



# Introduction to Mixed MPI and OpenMP

Cray XE6 Workshop  
February 7, 2011

David Turner  
NERSC User Services Group



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science



National Energy Research  
Scientific Computing Center



Lawrence Berkeley  
National Laboratory



# Parallel Models

- MPI describes parallelism *between* processes (with separate address spaces)
- OpenMPI provides a shared-memory model *within* a process
  - Loop level and parallel sections



# Hybrid MPI/OpenMP

- **Why?**
  - Good usage of shared memory system resource (memory, latency, and bandwidth).
  - Avoids the extra communication overhead with MPI within node.
  - Larger MPI messages.
  - Some problems/algorithms have multiple levels of parallelism



# Standards Interoperability

- **OpenMP**
  - If a thread is blocked by an operating system call (e.g. network I/O), the other threads remain runnable.
    - An MPI call like `MPI_Recv` or `MPI_Wait` only blocks the calling thread.
- **MPI**
  - More complex...



# MPI Thread Support

- **`MPI_Init_thread(requested, provided)`**
- **`MPI_THREAD_SINGLE`**
  - One thread in each process
- **`MPI_THREAD_FUNNELED`**
  - Only one thread makes MPI calls
- **`MPI_THREAD_SERIALIZED`**
  - Only one thread makes MPI calls at a time
- **`MPI_THREAD_MULTIPLE`**
  - Any thread may make MPI calls at any time



# MPI Threads in OpenMP Context

- **`MPI_THREAD_SINGLE`**
  - No OpenMP multithreading in the program
- **`MPI_THREAD_FUNNELED`**
  - All MPI calls made by master thread
    - Outside OpenMP parallel regions, or
    - Inside OpenMP master regions, or
    - Guarded by call to `MPI_Is_thread_main` MPI call  
(same thread that called `MPI_Init_thread`)



# MPI Threads in OpenMP Context

- **`MPI_THREAD_SERIALIZED`**

```
#pragma omp parallel  
...  
#pragma omp atomic  
{  
    ... MPI calls allowed here ...  
}
```

- **`MPI_THREAD_MULTIPLE`**

- Any thread may make an MPI call at *any* time
  - Link with `-lmpich_threadm`



## Threads and MPI-2 Standard

- Not required to support levels higher than **MPI\_THREAD\_SINGLE**
  - Not required to be thread safe in order to be standard-conforming
- Fully thread-compliant implementations will support **MPI\_THREAD\_MULTIPLE**
- A portable program that does not call **MPI\_Init\_thread** should assume that only **MPI\_THREAD\_SINGLE** is supported



## Current Situation

- All MPI implementations support **MPI\_THREAD\_SINGLE**
- Most MPI implementations support **MPI\_THREAD\_FUNNELED**
- Some MPI implementations support **MPI\_THREAD\_MULTIPLE**
- Many “easy” hybrid programs only need **MPI\_THREAD\_FUNNELED**
  - MPI used between loops parallelized with OpenMP



# A Hybrid Pseudo Code

```
program hybrid
  call MPI_INIT (ierr)
  call MPI_COMM_RANK ...
  call MPI_COMM_SIZE ...
  ... some computation and MPI communication
  call OMP_SET_NUM_THREADS(4)
  !$OMP PARALLEL DO PRIVATE(i) SHARED(n)
  do i=1,n
    ... computation
  enddo
  !$OMP END PARALLEL DO
  ... some computation and MPI communication
  call MPI_FINALIZE (ierr)
end
```



# Hopper Usage Overview

- **Compile as if “pure” OpenMP**
  - -mp=nonuma for PGI
  - -mp for Pathscale
  - -fopenmp for GNU
  - no options for Cray
  - Cray wrappers add MPI environment

```
#PBS -l mppwidth=48
setenv OMP_NUM_THREADS 6
aprun -n 8 -N 4 -d 6 ./a.out
```



# Useful aprun Options

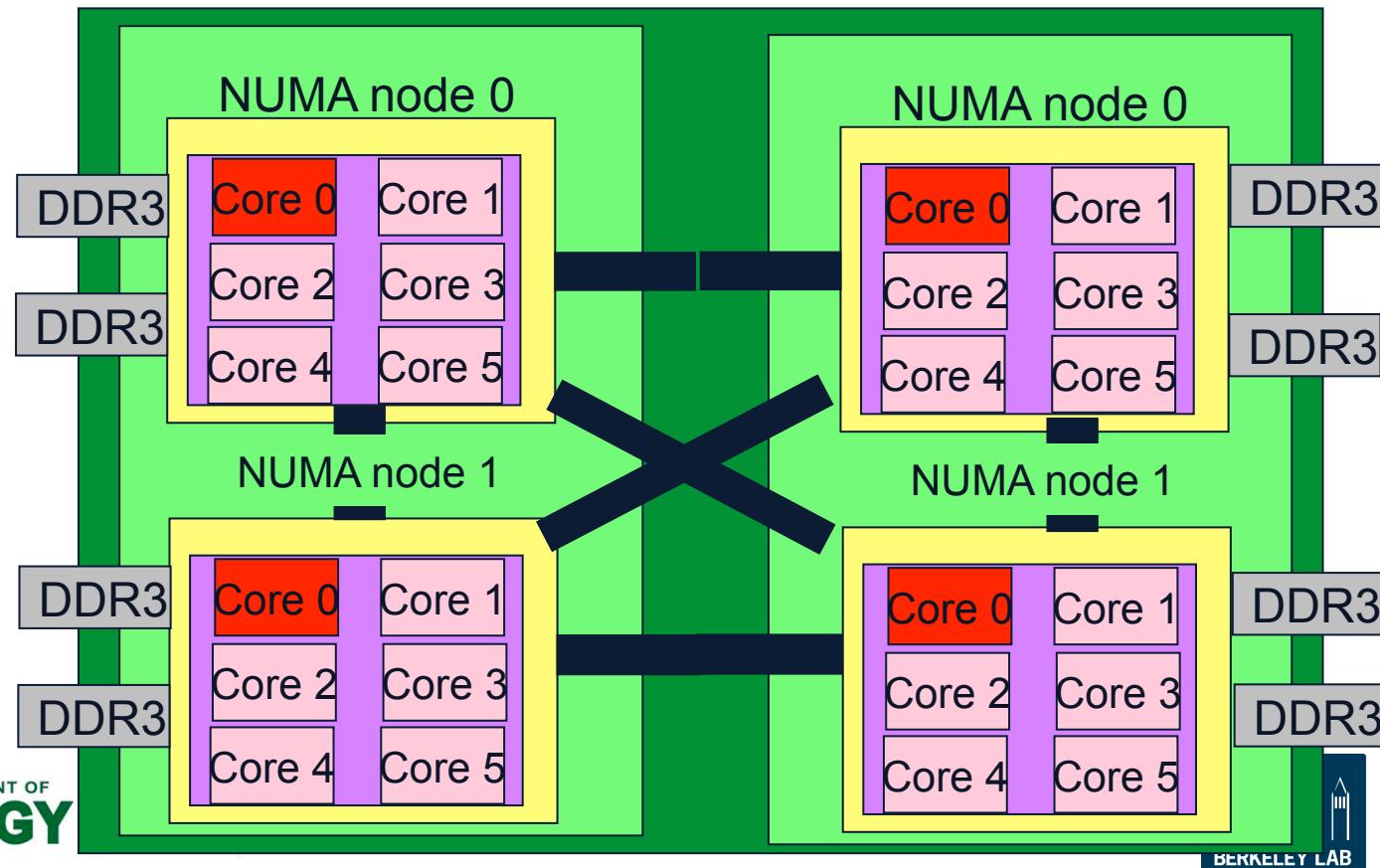
Option	Description
-n	Number of MPI tasks.
-N	(Optional) Number of tasks per Hopper Node. Default is 24.
-d	(Optional) Depth, or number of threads, per MPI task. Use <i>in addition to OMP_NUM_THREADS</i> . Values can be 1-24; values of 2-6 are recommended.
-S	(Optional) Number of tasks per NUMA node. Values can be 1-6; default 6
-sn	(Optional) Number of NUMA nodes to use per Hopper node. Values can be 1-4; default 4
-ss	(Optional) Demands strict memory containment per NUMA node; default is to allow remote NUMA node memory access.
-cc	(Optional) Controls how tasks are bound to cores and NUMA nodes. Recommendation for most codes is -cc cpu which restricts each task to run on a specific core.



# Hybrid MPI/OpenMP example on 6 nodes

- 24 MPI tasks with 6 OpenMP threads each

```
#PBS -l mppwidth=144  
setenv OMP_NUM_THREADS 6  
aprun -n 24 -N 4 -d 6 ./a.out
```



U.S. DEPARTMENT OF  
**ENERGY**

Lawrence Berkeley  
National Laboratory





# Controlling NUMA Placement

```
#PBS -l mppwidth=144 (so 6 nodes!)
```

- **1 MPI task per NUMA node with 6 threads**  
`setenv OMP_NUM_THREADS 6  
aprun -n 24 -N 4 -d 6 ./a.out`
- **2 MPI tasks per NUMA node with 4 threads**  
`setenv OMP_NUM_THREADS 4  
aprun -n 36 -N 8 -d 4 ./a.out`
- **3 MPI tasks per NUMA node with 3 threads**  
`setenv OMP_NUM_THREADS 3  
aprun -n 48 -N 12 -d 3 ./a.out`



U.S. DEPARTMENT OF  
**ENERGY**

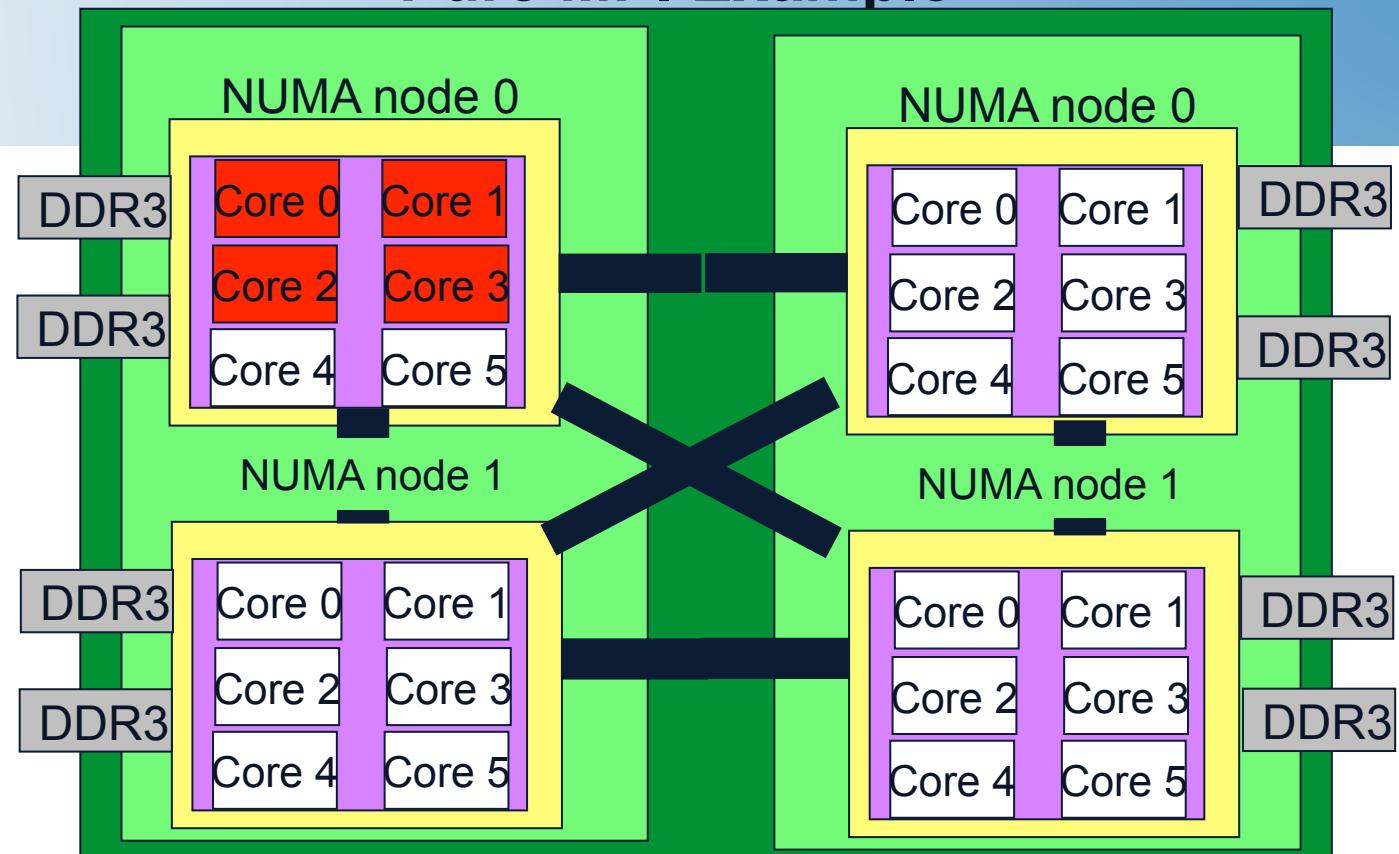
Office of  
Science



Lawrence Berkeley  
National Laboratory

## Pure MPI Example

- Example: 4 MPI tasks per node
- Default placement is not ideal when fewer than 24 cores per node are used.

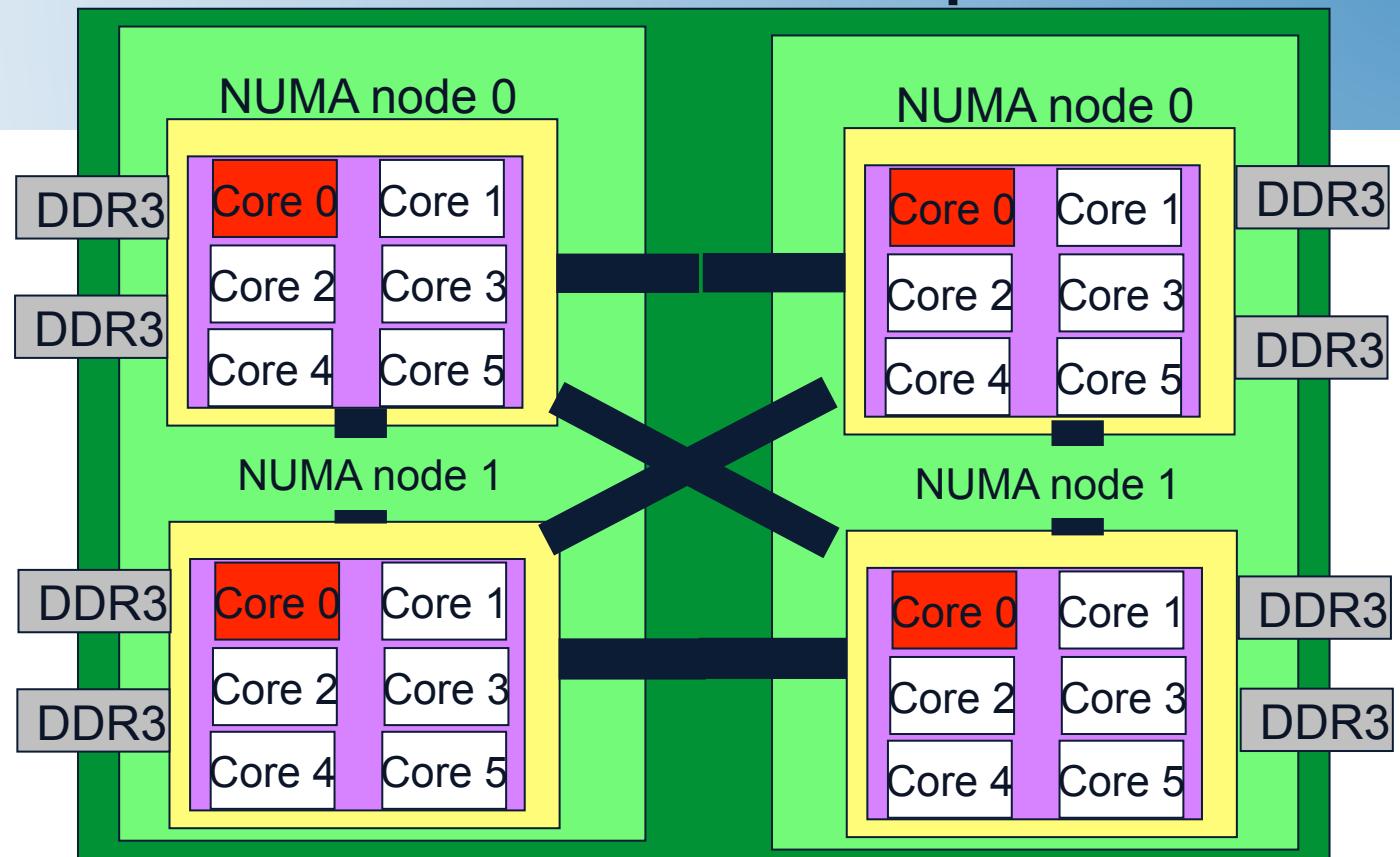


```
#PBS -l mppwidth=24  
#PBS -l walltime=00:10:00  
#PBS -N my_job  
#PBS -q debug  
#PBS -V
```

```
cd $PBS_O_WORKDIR  
aprun -n 4 ./parHelloWorld
```

# Better Pure MPI Example

- Example 4  
MPI tasks  
per node
- -S 1 flag  
says put on  
core on each  
NUMA node



```
#PBS -l mppwidth=24
#PBS -l walltime=00:10:00
#PBS -N my_job
#PBS -q debug
#PBS -V
```

```
cd $PBS_O_WORKDIR
aprun -n 4 -S 1 ./parHelloWorld
```



## Hands-On

**/project/projectdirs/training/XE6-feb-2011/Mixed**

**jacobi\_mpiomp.f90**

**jacobi\_mpiomp.pbs**

**indata**



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science



Lawrence Berkeley  
National Laboratory